REMARKS

The Examiner's objections have been carefully noted.

The final paragraph of the description on page 17 has been deleted because there are no alphabetic characters in the claims used to designate claim operations.

Amendments have been effected to claims 1, 7, 17, 18, 23 and 24 to address the informalities noted in paras. 2 and 4 of the Office Action.

In para. 5 of the Office Action, the Examiner has rejected claims 1-23 under 35 U.S.C. §101 as being directed to non-statutory subject matter because he construes the claims to be software, *per se*. We find this terminology '*per se*' very surprising and strangely reminiscent of European practice where Article 52(2)(c) EPC excludes 'schemes, rules and methods for performing mental acts, playing games or doing business, and *programs for computers*' from patentability. In order to compete in the marketplace of software-related IP, the EPO has been forced to construe this exclusion as meaning 'programs for computers *per se*' – but no such exclusion exists in US Patent Law and the Courts have repeatedly refused to limit the scope of what is patentable under 35 U.S.C. §101.

Thus, in Ex parte Lundgren, Appeal No. 2003-2088, Application 08/093,516, (Precedential BPAI opinion September 2005), the Board held that "there is currently no judicially recognized separate "technological arts" test to determine patent eligible subject matter under Sec. 101." [Lundgren at 9.]

> 35 U.S.C. § 101 provides: Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Thus there is no statutory bar under 35 U.S.C. §101 to software patents; nor is there any concept under US Patent Law as "software *per se*": it being reiterated that this is a purely European contrivance to justify the patentability of software inventions notwithstanding the statutory bar to computer programs under Article 52(2)(c) of the European Patent Convention!

It is further noted that while it is true that:

> The Supreme Court has ". . . recognized limits to § 101 and every discovery is not embraced within the statutory terms. Excluded from such patent protection are laws of nature, physical phenomena and abstract ideas." [Diamond v. Diehr, 450 U.S. 175,185, 209 USPQ 1, 7 (1981).]

However, the Examiner has not taken the position that claim 1 is directed to a law of nature, physical phenomena or an abstract idea, the judicially recognized exceptions to date to § 101. Rather, the Examiner has implicitly found a separate "technological arts" test in the law (namely "software *per se*") and has determined that claim 1 does not meet this separate test.

It is further noted that in above-referenced Ex parte Lundgren, the Board held:

> Our determination is that there is currently no judicially recognized separate "technological arts" test to determine patent eligible subject matter under § 101. We decline to propose to create one. Therefore, it is apparent that the examiner's rejection can not be sustained.

It is respectfully submitted that there is no basis in the Law for the Examiner's rejection under 35 U.S.C. §101 and it is accordingly urged that the objection be withdrawn.

Nevertheless, Applicant is aware that it is well-established that the USPTO Guidelines do not follow the majority decisions of the Board of Appeals in this regard and follow instead those dissenting voices among the judiciary that hold:

> ... computer program *per se* is not statutory subject matter because it does not fall within any statutory class. See In re Chatfield,545 F.2d 152, 159, 191 USPQ 730, 737 (CCPA 1976) (Rich, J., **dissenting**)

Claim 1 has therefore been amended to introduce the hardware features that the framework is operated by a management server and that the user interface requires a console. With this amendment, the claim defines an interaction between hardware and software and, as such, is clearly statutory under 35 U.S.C. §101. In this connection the Guidelines state:

> When a computer program is recited in conjunction with a physical structure, such as a computer memory, USPTO personnel should treat the claim as a product claim.

Likewise, independent claims 22 and 24 have been amended to recite *a machine-readable memory storing an object-oriented data structure for automating monitoring and configuration of an application complex* and *a machine-readable memory storing a plugin including* [such an] *object-oriented data structure*, respectively. As such, these claims clearly meet the Guidelines as well as being allowable under 35 U.S.C. §101.

In para. 6 of the Office Action, the Examiner has rejected claim 1-24 under 35 U.S.C. §103(a) as being unpatentable over Carlson (US Pat. No. 6,697,849) in view of Choquier *et al.* (US Pat. No. 5,951,694). The objection is respectfully traversed.

In order to understand better why it is submitted that the claims are patentable over Carlson and Choquier *et al.* some brief background to the invention will be helpful.

The object of the invention is to provide a generic framework that allows management control of multiple tiers of servers that are dedicated to different applications and whose architecture and functionality is not necessarily fixed or known in advance. Moreover, the framework allows creation and population of new application-complex types and provides a graphical user interface that is appropriate

to the new application-complex type. Thus, for example, referring to Fig. 2, the GUI shows the architecture of each application complex, namely SALES and SUPPORTS in the specific example shown in the figure and the tree hierarchy likewise shows the architecture of each application complex, allowing their respective configurations to be changed by dragging and dropping servers.

It is true that Carlson also shows in Fig. 11 a user interface of a partial tree view of application servers in an application server cluster that may appear to be similar to Fig. 2 of the present application. However, any similarity is superficial since the invention is not defined by the use of a tree hierarchy to manage an application complex but rather by the use of a plugin that is used by a generic framework to configure the appropriate tree hierarchy for each application complex. Carlson provides no teaching to do this: his tree hierarchy relates only to a single application complex of predefined known type. Carlson provides no teaching to add different application complexes even of the same type, and certainly not of different types.

To this end, the invention as claimed is directed to a computer-implemented framework for managing application complexes, each application complex comprising multiple tiers of servers, where servers in a common tier run an identical application, and all the servers work together to provide a specific service. The application complex is definable via an application complex type.

Already in the preamble to the main claim, the invention as claimed is distinguished over Carlson in that Carlson relates to only a single application complex type. Thus Carlson does not teach or suggest an application complex being definable via an application complex type.

Thus, Carlson shows in Figs. 2A-2C a single web servers 104 coupled to a single tier of application servers 108A, 108B commonly connected to a database 110. Carlson provides a mechanism for managing the application servers 108A, 108B. Thus, as noted in col. 3:

> ... it is apparent that application servers may integrate a diverse range of services, where these services may interact with many different types of servers, systems, or other services. For example, an application server may act as a platform hub connecting web servers, database servers/services, e-commerce servers/services, legacy systems, or any of various other types of systems or services. A key benefit of many application servers is that they not only provide this service/system integration, but typically also provide centralized administrative or management tools for performing various aspects of system and application administration. [Col. 3, lines 40-51]

> Application servers may also provide tools for easily configuring the application to utilize various of the application server services described above. For example, administrators may use a tool to set the result caching criteria for particular application components or pages, or may use a tool to specify which documents to index or to specify indexing methods, etc. [Col. 3, lines 58-65]

> Application servers may provide tools for dynamically managing various factors affecting application performance, e.g. by adjusting the application services and support features described above. For example, application server tools may allow administrators to: dynamically adjust the number of database connections maintained in a database pool, in order to determine the optimum pool size for maximum performance clear or resize application output caches dynamically change various aspects of system or application security schedule or trigger events, such as events for sending e-mail reports to application users, generating reports based on collected data, etc. start and stop various application services, such as email or FTP services, from a centralized user interface. [Col. 4, lines 1-65]

Thus, it is agreed that Carlson teaches a system that provides tools for configuring an application complex. However, there is no suggestion in Carlson to allow multiple application complexes of different types to be managed by a generic framework that recognizes different application complex types via a plugin that is coupled to the framework and contains information allowing the framework to provide a user interface that displays objects such as tiers of the **new** application complex type and servers in different tiers thereof so as to allow configuration of servers in the **new** application complex type via the respective plugin.

Rather, as noted above, Carlson relates to an application complex specific to the domain of Web serving and having no capability to recognize and allow configuration of other application complex types.

The Examiner avers that the feature in claim 1 that the "application complex being definable via an application complex type" is taught by Carlson col. 2 lines 19-28; col. 3, lines 38-51 and col. 4, lines 21-58. But as noted above, these sections merely explain what is meant by *application services* and *application servers*. Moreover, it is clear that while Carlson attempts to define *application servers* fairly broadly his description relates only to web-based applications. Thus, he states at col. 3, lines 1-37:

> Many web applications need to perform various types of searching or indexing operations. Application servers may also provide application services for indexing or searching various types of documents, databases, etc.
>
> As noted above, many web applications may perform various types of complex, multi-step transactions. Application servers may also provide support for managing these application transactions. For example, this support may be provided via a software component model supported by the application server, such as the Enterprise JavaBeans™ component model, or via integration with third-party transaction process monitors, etc.
>
> It is often desirable to enable web applications to perform certain operations independently, as opposed to in response to a user request. For example, it may be desirable for an application to automatically send a newsletter to users via email at regularly scheduled intervals. Application servers may support the creation and scheduling of events to perform various types of operations.

> Many types of web applications need to perform e-commerce transactions, such as credit card transactions, financial data exchange, etc. Application servers may provide services for performing various types of e-commerce transactions or may provide an integrated third-party e-commerce package for applications to use.
>
> Web applications often need to utilize various types of standard network application services, such as an email service, FTP service, etc. Application servers may provide these types of services and may enable applications to easily integrate with the services.
>
> Web applications often need to log various conditions or events. Application servers may provide an integrated logging service for web applications to use.

It is to be noted that in fact all of these examples are web-based and Carlson appears not to describe even a single tier of applications servers that run a non Java web-based application. Likewise, the management tools mentioned by Carlson at col. 3 lines 38-51 relate to web-based computing services that may be provided by application servers, or administration tools that may be provided by them. This also is unrelated to generic management framework for application complexes.

But far more important even than this is the fact that Carlson does not provide a generic framework that can be updated via appropriate plugins to manage different types of application complex that were not predefined but are made known to the framework via the plugin that encapsulates a relationship between disparate resources composing the respective application complex type and respective characteristics of said resources.

Similarly, the description at col. 4 lines 21-58 relates to clustering of application servers and not to a generic management framework for application complexes: at the closest, a clustering technology might be used in some specific type of application complex but it does not allow for managing different types of application complex.

We therefore respectfully maintain that Carlson does not relate to a system as defined by the preamble of the independent claims 1, 22 and 24.

As noted above, claim 1 recites:

> a plugin interface adapted for connection to the framework of a plugin in respect of each application complex type, wherein said plugin encapsulates a relationship between disparate resources composing the respective application complex type and respective characteristics of said resources

The Examiner contends that this feature is taught by Carlson Fig. 4; col. 3, lines 38-51 and col. 10, lines 7-32. We have already related to the description at col. 3, lines 38-51 and respectfully dispute the Examiner's contention that it discloses a "plugin interface" as claimed.

Fig. 4 of Carlson does disclose what he refers to as a "plugin" but his plugin is certainly not of the kind to which the present invention relates and is described in claim 1 of the present application. Thus, referring to col. 10, lines 7-32 (as noted by the Examiner), Carlson states:

> For example, FIG. 4 illustrates a web server client 240 which **comprises a standard web server extension or plug-in 242. The web server plug-in 242 may be any of various well-known types of plug-ins enabling web servers to communicate with other systems**, including NSAPI, ISAPI, optimized CGI, etc. As shown, the protocol manager service 220 includes "listener" modules or components, e.g. an NSAPI listener, ISAPI listener, etc., for communicating with the web server plug-in. The listener modules may communicate with the web server plug-in via the standard HTTP or HTTPS protocols. [Emphasis added]

Carlson's plug-in may be adapted to perform load balancing as described at length at col. 11, lines 12*ff* but it remains a conventional plugin that merely serves to enable communication between a web server and other systems. Carlson certainly does not suggest the use of a new type of plugin that *encapsulates a relationship between disparate resources composing the respective application complex type and respective characteristics of said resources* as recited in claim 1 of the present application. To the contrary, there is no suggestion that his plug-ins are new *per se*: indeed as noted above, Carlson states at col. 10, lines 11-14 that his plug-in is **standard** and **known**. Moreover, load balancing *per se* is certainly well-known in the art. In Carlson, different application servers can be pulled into and out of operation depending on load. Since, all application servers in Carlson are, in fact, dedicated to the same type of application complex, his web server employs a common plug-in that serves to interface between the web server and the application servers.

Carlson relates extensively to load balancing but to no other use of his plug-in. He also discusses at col. 16 what he refers to as "graceful distribution" – but in all cases his plug-in allows load balancing between the application servers of a single type of application. There is specifically no teaching in Carlson to provide a plug-in that informs a generic framework of the composition of a new application complex type, to allow the generic framework to create a GUI based on the new application complex type and that allows a user to use the GUI to populate servers of the new application complex type.

Thus, claim 1 as amended recites:

> a user interface coupled to the management server and operating under control of the interface for providing general tasks that are independent of operational semantics of the application complex and that is responsive to user operations input to said framework via a console for interfacing with the framework for defining an instance of the application complex and allowing [[the]] a user to populate the application complex with servers.

The Examiner contends that this feature is taught by Carlson col. 2, lines 19-34; col. 4, lines 21-28; col. 13, lines 23-29. We have already related at length to cols. 2 and 4. Col. 13, relating to FIGS. 10-12, which describes:

> FIGS. 10-12 illustrate an exemplary user interface of an administrative tool for adjusting load balancing factors such as those described above. FIG. 10 illustrates a user interface screen for setting server load criteria values, such as those shown in the FIG. 8 table. Administrators may adjust the weight for each factor as appropriate, in order to maximize performance for a particular application server.

It appears that the Examiner has cited this feature because it describes a user interface. But Carlson's user interface is not defined by a generic framework that is common to **different application complexes** each having a respective plugin to the framework that allows the framework to create a new instance of the application complex and to create an appropriate user interface for managing the application complex.

The Examiner further refers to Choquier col. 7 lines 44-62. In the invention as claimed in claim 1, populating an application complex with servers is in the context of a generic management framework for application complexes. Claim 1 of the present application defines the interaction between the management framework and the plugin that supports the application complex type. When a server is added to the application complex, the plugin is called by the management framework to do all the required configurations on the server and possibly also on other servers of the application complex (see also claim 7 and page 7 lines 23-26 of the application). This enables inter-server and inter-tier configurations that might be required for a specific application complex type when a new server is added, where all the knowledge of the required inter-server and inter-tier configuration is encapsulated only in the plugin. Thus support for an application complex that requires such involved level of inter-server and inter-tier configurations for each new server that is added can be included within the generic management framework by providing the supporting plugin (that interacts with the management framework according the Configuration-Provider plugin interface as recited in claim 1 and in Section 2. "Configuration Provider Interface" starting on page 7, line 29 of the application).

In Choquier, there is no notion of generic management framework and plugins for service groups. In contrast, the management framework itself performs the configuration when a new server is added - by activating a DLL on the server [see col. 17, lines 19-25; col. 23, lines 49-52], configuring an Arbiter and updating a service map. Thus it is not possible to plug in an application complex that requires a more involved level of inter-server and inter-tier configurations per server addition than the 'fixed' configuration steps done by the framework for any type of service group.

Claim 2 of the application adds the limitation that:

> the plugin is adapted to convey to the framework information relating to the type of the application complex, the number of tiers, the application which the servers in each tier should run, and one or more

properties of the application complex whose values can be specified by the user for each instance of the application complex type.

The Examiner contends that claim 2 is rendered obvious by the Figs. 11 and 14 in Carlson and the disclosure at col. 15, lines 29-27, which reads:

> FIG. 14 illustrates an exemplary user interface of a tool for enabling administrators to specify sticky load balancing for certain application components. FIG. 14 illustrates a group of application components which, for example, may be displayed by navigating through a hierarchy tree such as shown in FIG. 11. The "Sticky LB" column of the user interface has a checkbox allowing sticky load balancing to be turned on for particular application components.

It is respectfully submitted that there is nothing in Figs. 11 and 14 of Carlson that teaches the features of claim 2, either on its own and *a fortiori* when taken in combination with claim 1. Specifically, the hierarchy tree shown in Fig. 11 and described at col. 13, lines 29-41 relates to a **single** application complex type that is **predefined** and **whose hierarchy is therefore known to the GUI**. As noted above, Carlson's plugin is merely a standard interface that serves to enable communication between a web server and other systems. Carlson does not teach a plugin that is *adapted to convey to the framework information relating to the type of the application complex, the number of tiers, the application which the servers in each tier should run, and one or more properties of the application complex whose values can be specified by the user for each instance of the application complex type.*

The Examiner avers that Carlson teaches in FIG. 11 and col. 6, lines 32-33 a server in a hierarchy, including identifiers that indicate types and that it would have been obvious to include the tiers (where the servers are located) in the higher levels of the tree and that by listing the elements of the system, including tiers, it shows how many tiers exist. It is respectfully submitted that these statements reflect a misunderstanding on the Examiner's part that hopefully will now have been clarified in the light of the above explanations. However, for the sake of record, it should be noted that the invention as claimed in claim 2 operates in exactly the **opposite** way to that apparently understood by the Examiner. Thus, it is not that the number of tiers in the application complex and the number of servers in each tier is, or can be, inferred from the hierarchy tree as suggested by the Examiner; rather what the invention teaches and claims is that based on a predefined configuration of the application complex type as determined from the plugin, the framework instructs the GUI how to represent the hierarchy tree for each application complex type. Likewise, it is not that the identifiers (we assume by this the Examiner means NAS1 and EJB in FIG. 11 of Carlson) depict different application complex types as apparently suggested by the Examiner (in fact they don't: rather they depict different tiers of the same application complex); but rather that the framework instructs the GUI how to represent the hierarchy tree for each application complex type from the information contained in the plugin associated with each application complex type.

Carlson neither relates to multiple application complex types, nor is his GUI built on-the-fly under instruction from a generic framework based on definitions of

different application complex types. It is therefore respectfully submitted that claim 2 is allowable.

Claim 3 of the application adds the limitation that:

> the plugin is responsive to a change in one or more properties of the application complex for configuring at least one of said servers in accordance with said change.

The Examiner avers that this claim is rendered obvious by the combination of Carlson and Choquier. Thus, the Examiner refers to col. 4, lines 1-38 and col. 13, lines 23-41 of Carlson. No specific reference to Choquier is made. The sections of Carslon to which the Examiner refers describe a user interface for adjusting load balancing factors and setting server load criteria. No reference is made to a user interface for a generic management framework of application complexes.

Claim 4 of the application adds the limitation that:

> the plugin is adapted to convey to the framework information relating to one or more properties of the application complex whose values are to be monitored by the plugin and the plugin is adapted to monitor said properties and return their respective values or functions thereof to the framework.

The Examiner contends that claims 4, 5 and 6 are rendered obvious by Carlson at col. 11, lines 12-42 and col. 12, lines 6-26, all of which relate to a load balancer that is included in a Web-server plug-in to a Web-server that affects Web communication. Carlson does not describe a plug-in into a management framework for application complexes that adds a capability to the framework to support another type of application complex and performs monitoring of some properties of the application complex as defined by claims 4, 5 and 6 of the present application.

The same applies to claims 7 and 8. Claim 7 adds the limitation that:

> the plugin is responsive to a new server being added to a tier in the application complex for automatically configuring the new server and any other servers in the application complex that relate to the new server.

The Examiner has rejected claims 7 and 8 based apparently on Carlson col. 7, lines 53-62 and col. 11, line 58 to col. 12, line 7. Since this description does not appear to relate to the subject matter of claims 7 and 8, we assume that the Examiner meant to refer to Choquier and not Carson.

Choquier describes at col. 7 lines 53-62 the allocation of additional servers to a heavily-loaded service. At col. 11 line 58 through col. 12 line 7 he also relates to use of the service map 136 to inform the Gateways 126 of the additional, deletion or change in state of a server 120 in the system. In this case, we refer the Examiner to our previous comments regarding Choquier in connection with claim 1. Specifically, Choquier makes no suggestion of a generic management framework and plugins for service groups. Rather, the management framework itself performs the configuration when a new server is added - by activating a DLL on the server [see col. 17, lines

19-25; col. 23, lines 49-52], configuring an Arbiter and updating a service map. Thus it is not possible to plug in an application complex that requires a more involved level of inter-server and inter-tier configurations per server addition than the 'fixed' configuration steps done by the framework for any type of service group.

The same applies to claims 9 and 10, which also stand rejected in view of Carlson in combination Choquier. Thus, Carlson at col. 4 lines 33-38 describes adding application servers to clusters in general. Choquier col. 7 lines 53-62, and col. 11, line 58 through col. 12 line 7 relates to adding servers to a loaded service. Nothing in either Carlson or Choquier taken singly or in combination teaches the limitations of claim 9 of the present application, which relates to a generic management framework for application complexes as recited in claim 1, discloses the plugin adaptation to request the management framework for additional server, whereby the plugin is capable of deep understanding of the state of the specific application complex it supports, and thus has a better knowledge as to when to request additional server (or to remove a server). In contrast, the generic measurements performed by Choquier's framework relate mainly to CPU utilization, which are measured for each service.

Claims 11 to 17 and 19 to 21 add various limitations relating to the interface and are believed to be allowable by virtue of their being dependent on allowable base claims and for the detailed reasons provided above relating to Choquier.

Claim 17 has been amended to overcome the objection raised in para. 4 of the Office Action under 35 U.S.C. § 112, second paragraph. It is thus now clarified that the first and second instances are different. For the sake of completeness, it is also stated that the respective tier in each of said instances has an identical class. This must be the case to allow interchangeability of servers between tiers. No new matter is thereby added and the Examiner is referred to page 10, lines 12-15 of the application as filed, relating to the Class named 'TierDefintion.'

In order to avoid a contradiction between claim 1 and 17, the statement in claim 1 that "servers in different tiers run different applications" has been deleted from claim 1. This statement is clearly incompatible with claim 17, which requires that servers in different tiers run the same application and is in any case redundant.

Claim 18 of the application has been re-cast as dependent on claim 1 so as to avoid any apparent conflict with claim 17 and relates to the situation where a single application complex instance has two or more tiers having the same TierDefintion class and it is desired to move servers between tiers.

The Examiner has rejected this claim based on Carlson, col. 7, line 60-col. 8, line 7 and specifically to the combination of Carlson, col. 7, lines 60–63 in combination with Choquier col. 23, lines 36-48. This objection is likewise traversed in view of our previous comments relating to Choquier. In addition, it is reiterated that neither Carlson nor Choquier teaches or suggests an application complex type whose definition is contained within a plugin that is connected to a generic framework, thus allowing a user to create multiple instances of the same application complex type via a user interface that is responsive to instructions by the framework.

Claims 22 to 24 are believed to be allowable for the same reasons as provided previously. Claim 22 recites an object-oriented data structure within the context of the management framework for application complexes and is therefore deemed allowable for the reason stated above. Specifically, neither Carlson nor Choquier relates to an:

> application complex being definable via an application complex type and said object-oriented data structure comprising objects that encapsulate a relationship between disparate resources composing respective application complex types and respective characteristics of said resources thus allowing an instance of an application complex type to be defined.

In view of the amendments to the independent claims and the above remarks, it is respectfully submitted that the application is suitable for allowance and favorable reconsideration is accordingly requested.

Please charge any fees associated with this paper to deposit account No. 09-0468.

Respectfully submitted,

By: /Stephen C. Kaufman/
Stephen C. Kaufman
Reg. No. 29,551
Phone No. (914) 945-3197

Date: January 3, 2007

IBM Corporation
Intellectual Property Law Dept.
P. O. Box 218
Yorktown Heights, New York 10598